27/11/24

LAB - 10

Q) Demonstrate inter process Communication and deadlock.

```
class Q {
 int n;
 boolean value Set = false;
 synchronized int get () {
 while (! value Set)
  try {
    System. out. println ("In Consumer Waiting \n");
    wait();
   }
   catch (InterruptedException e) {
   System. out. print ln (" InterruptedException caught");
   }
 System. out. println ("Got : " + n);
 value Set = false;
 System. out. print ln ("In Intimate Producer \n");
 notify ();
 return n;
}
synchronized void put (int n) {
  while (value Set)
  try {
   System. out. print ln (" Producer Waiting");
   wait ();
  }
  catch (Interrupted Exception e) {
   System. out. print ln ("InterruptedException caught");
  }
this.n = n;
valueSet = true;
System.out.println("Put :" + n);
System.out.println("Intimate Customer");
notify(); }
}


class Producer ~~Procedure~~ implements Runnable {
Q q;
~~Procedure~~
 Producer(Q, q) {
    this.q = q;
    new Threads(this, "Producer").start();
 }
 public void run() {
    int i = 0;
    while (i < 15) {
    q.put(i++); }
    }
}


class Consumer implements Runnable {
  Q q;
  Consumer (Q, q) {
     this.q = q;
     new Thread(this, "Consumer").start();
  }
```

```
public void run () {
  int i = 0 ;
  while (i <15) {
    int r = q. get ();
   System. out. print ln (" consumed !" + r);
   i ++ ; }
  }
}

class PC Fixed {
  public static void main (String args [ ]) {
   Q q = new Q ();
   new Producer (q);
   new Consumer (q);
   System. out. print ln ("Press Control-C to stop");
  }
}
```

Put : 1

Get : 1

Put : 2

Get : 2

Put : 3

Get : 3

Put : 4

Get : 4

Put : 5

Get : 5

27/11/24

Deadlock

```
class A {
 synchronized void foo (B b) {
 String name = Thread. currentThread(). getName();
 System. out. println (name + " entered A. foo");
 try {
  Thread. sleep (1000); }
  catch (Exception e) {
   System. out. println (" A Interrupted");
  }
 System. out. println (name + " trying to call
 B. last()");
 b. last();
 }
 void last() {
  System. out. println(" Inside A. last"); }
}

class B {
 synchronized void bar (A a) {
  String name = Thread. currentThread(). getName();
  System. out. println (name + " entered B. bar");
  try {
    Thread. sleep (1000); }
  catch ( Exception e) {
  System. out. println (" B Interrupted");
  }
System. out. println (name + " trying to call A.last()");
a. last() ;
}

void last () {
 System. out. println ("Inside A. last"); }
}
class Deadlock implements Runnable {
   A a  = new A ();
   B b   = new B ();
   Deadlock () {
    Thread. currentThread(). setName ("MainThread");
    Thread t  = new Thread ( this , "Racing Thread");
    t. start () ;
    a. foo (b) ;
   System. out. println ("Back in main thread"); }

public void run () {
  b. bar (a) ;
  System. out. println ("Back in other thread");
  }
  public static void main ( String args []) {
     new Deadlock (); }
}
```

Output :

```
Racing Threads entered B.bar
MainThread entered A. foo
Main Thread trying to call B. last ()
Inside A. last
Back in main thread
Racing Thread trying to call A. last ()
Inside A. last
Back in other thread
```

Seen

[illegible]
27/4/24